VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2015 Tomáš Špacír

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Špacír**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: GIRITON Systems s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

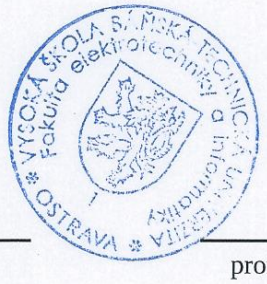
Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Kožusznik, Ph.D.**

Konzultant bakalářské práce: Ing. Jan Gřeš, MSc.

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 2015 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2015 .....

Rád bych na tomto místě poděkoval především mému mentorovi Ing. Janu Gřešovi za rady, které mi poskytl při plnění pracovních úkolů, a také za umožnění vykonávat odbornou praxi právě ve firmě Giriton Systems. V neposlední řadě bych chtěl poděkovat i mým spolupracovníkům, kteří byli vždy ochotni poradit a nasměrovat mě ke správnému řešení. Nakonec bych chtěl poděkovat i mému vedoucímu práce panu Ing. Janu Kožusznikovi za rady a připomínky k mé bakalářské práci.

Abstrakt

Cílem této bakalářské práce je popis činnosti, kterou jsem vykonával v rámci odborné praxe ve firmě GIRITON Systems s.r.o. V první části práce charakterizuji firmu a její zaměření. Dále popisuji technologie, se kterými jsem se seznámil a které jsem využil pro plnění pracovních povinností. Ve třetí části se zabývám zadanými úkoly a jejich řešením. V závěru hodnotím své působení ve firmě a nabyté znalosti.

Klíčová slova: GIRITON Systems s.r.o, Java, Vaadin

Abstract

The aim of this bachelor thesis is a description of the activity which I performed in the context of professional practice in GIRITON Systems Ltd. In the first part I describe the company and its specialization. I also describe the technology that I have met and I used it to fulfill work duties. In the third section I deal with specified tasks and their solutions. In the end I appreciate my activity in the company and acquired knowledge.

Keywords: GIRITON Systems Ltd, Java, Vaadin

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
CSS	–	Cascading Style Sheets
GWT	–	Google Webmasters Tools
HTML	–	Hyper Text Markup Language
IS	–	Informační Systém
JNA	–	Java Native Access
JNI	–	Java Native Interface
RFID	–	Radio Frequency Identification
SQL	–	Structured Query Language
XPath	–	XML Path Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Při studiu na Vysoké škole Báňské jsem se v každém semestru setkal s nějakou novou technologií či novým programovacím jazykem, však až ve třetím semestru mě poprvé velmi zaujal programovací jazyk Java. Po odevzdání semestrálního projektu jsem se ve svém volném čase pokoušel s tímto jazykem více seznámit, a proto jsem naprogramoval své menší projekty jako např. "hledání min". Když jsem se o semestr později dozvěděl, že mohu absolvovat v rámci bakalářské práce stáž u firmy, neváhal jsem. Hlavním důvodem byly bohaté zkušenosti, které jsem chtěl získat a využít své znalosti, které jsem v průběhu studia nabyl. Při hledání stáží jsem si vybíral především firmy zabývající se vývojem aplikací v programovacém jazyce Java. Nejvíce mne zaujala firma GIRITON Systems, kde jsem byl po absolvování pohovoru přijat.

V této práci se nejprve zabývám představením firmy GIRITON Systems s.r.o. a svým pracovním zařazením v této firmě. Dále se věnuji technologiím a nástrojům, se kterými jsem se za dobu své stáže seznámil. Uvádím zde i popis zadaných úkolů a jejich následné řešení. Práce také obsahuje zhodnocení mých scházejících znalostí v průběhu praxe. V závěru práce hodnotím své získané zkušenosti.

2 GIRITON Systems s.r.o.

GIRITON Systems s.r.o je mladá rozvíjející se firma, která vznikla v roce 2011 pod vedením Jana Gřeše MSc. Firma sídlí v moderní budově podnikatelského inkubátoru v areálu Vysoké Školy Báňské - Technické Univerzity Ostrava. [1]

2.1 Představení firmy

Firma se zabývá vývojem podnikových informačních systémů se zaměřením na moderní cloudové systémy a jejich propojení s tablety a mobilními telefony.

Obrázek 1: Logo firmy.

2.2 Docházkový systém

Docházkový systém umožňuje sledovat, spravovat či vyhodnocovat docházku zaměstnanců. Evidovaná data jsou sbírána pomocí mobilních terminálů (tabletů nebo mobilních telefonů) jak uvnitř budov, tak i v terénu a pomocí bezdrátové technologie přenášena do informačního systému, kde se vyhodnocují. Zákazník má tak možnost přistupovat k aktuálním informacím odkudkoli z internetu. [1]

2.3 Výrobní systém

IS GIRITON umožňuje také sledování a řízení výroby v malých a středních firmách výrobního charakteru. Díky terminálům, které jsou tvořeny tablety či mobilními telefony, se zaznamenávají právě prováděné operace určitým zaměstnancem. Tyto operace se identifikují podle RFID čteček a čárových kódů. Pomocí systému je prakticky online viditelné množství materiálu na skladu. [1]

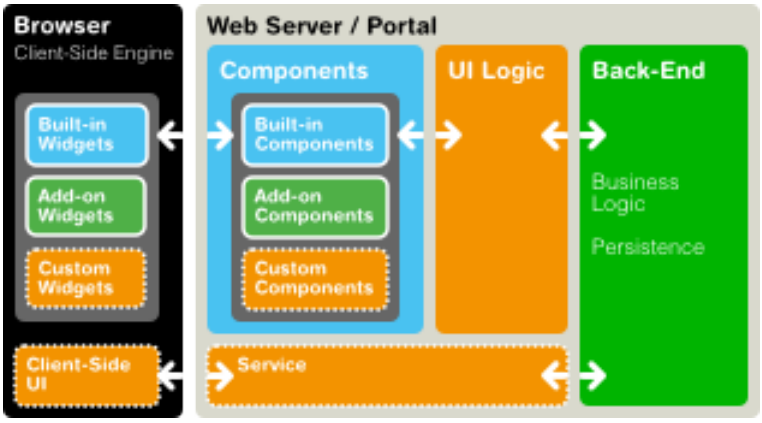
2.4 Pracovní pozice a její náplň

Ve firmě GIRITON Systems jsem zaujímal pozici Java programátora. V průběhu praxe jsem si vyzkoušel práci v různých částech informačního systému GIRITON. Ze začátku jsem pracoval na aplikacích pro správu baterie notebooku nebo tisku pomocí Adobe-Readeru. Později jsem se dostal k IS GIRITON od návrhu a implementace grafického uživatelského rozhraní až po testování systému.

3 Použité technologie

3.1 Vaadin

Vaadin je framework používající se pro tvorbu moderních webových aplikací v jazyce Java a následně pomocí GWT převeden na Javascript. Ten je poté spuštěn ve webovém prohlížeči. Programátor tudíž nemusí mít žádné znalosti HTML, CSS či Javasriptu a i přesto může vytvořit webové uživatelské rozhraní vysoké kvality. Architektura webové aplikace vytvořené ve Vaadin frameworku je patrná na obrázku 2. Při popisu Vaadin frameworku a jeho součástí používám informace z knihy Book of Vaadin. [2]

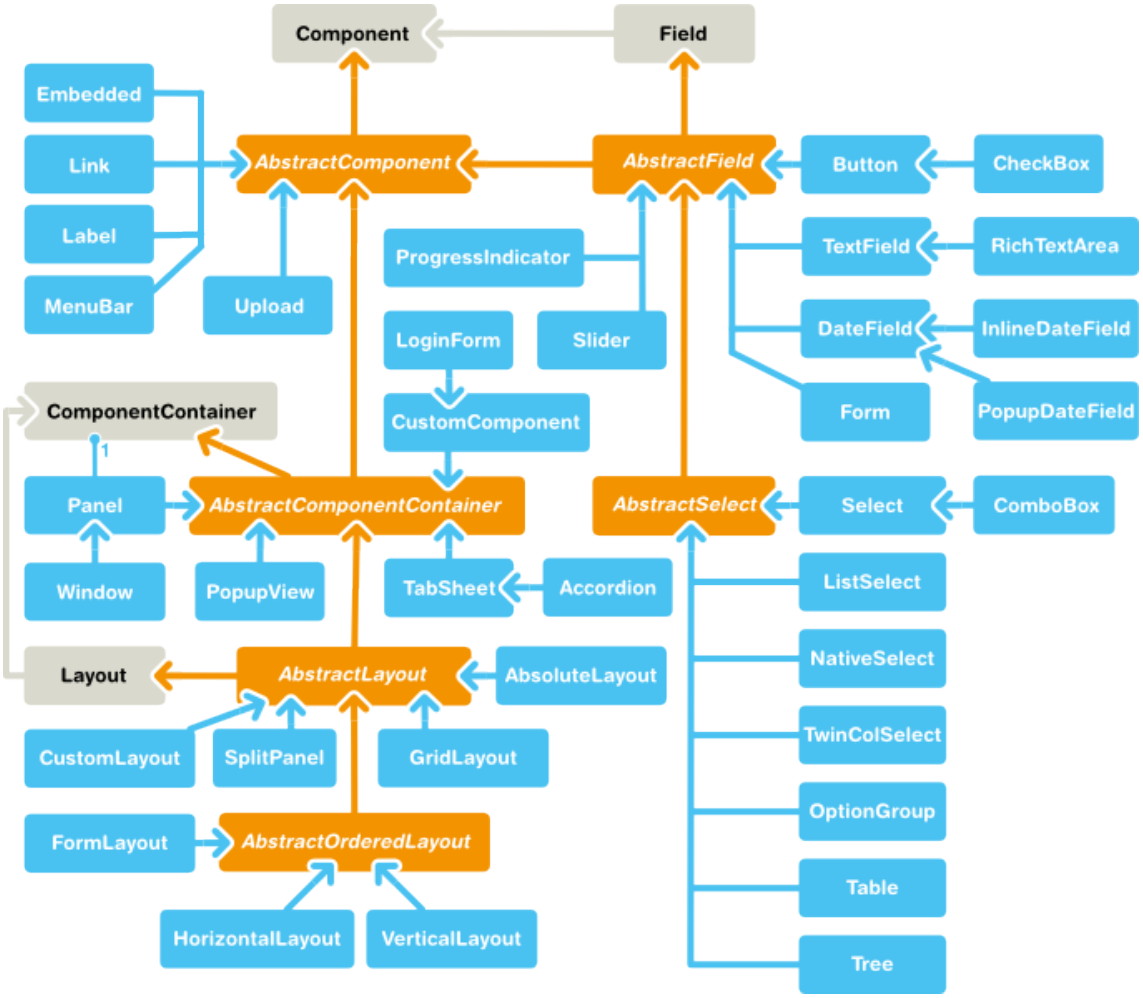


Obrázek 2: Architektura webové aplikace vytvořená ve Vaadinu. [2]

Tato architektura se skládá ze server-side frameworku a client-side enginu. Client-side engine funguje uvnitř prohlížeče jako Javascript, který vykresluje uživatelské rozhraní, a veškeré provedené změny jsou zaslány na server pomocí AJAXu, kde jsou zpracovány.
Vaadin poskytuje ucelenou sadu komponent uživatelského rozhraní a dovoluje definovat vlastní komponenty. Přehled těchto komponent je na obrázku 3. Právě na tomto obrázku jsou rozhraní vyznačeny šedě, abstraktní třídy oranžově a běžné třídy modře. Komponenty lze rozdělit do dvou skupin:

1. Skupina komponent, která slouží pro interakci s uživatelem.
2. Skupina komponent, které rozšiřují rozhraní *ComponentContainer*. Tyto layout komponenty mohou obsahovat komponenty jiné a mohou být umístěny na konkrétní místa v uživatelském rozhraní.

Mimo standartních layout komponent lze ve Vaadinu použít třídu *Customlayout*, pomocí které lze definovat vlastní umístění jakýchkoliv obsažených komponent pomocí HTML šablony.



Obrázek 3: Hierarchie Vaadin komponent.

3.2 Selenium

Selenium je nástroj pro automatizaci testů, které je možno spouštět v mnoha prohlížečích a napříč mnoha platformami. Umožňuje uživateli kontrolu nad webovým prohlížečem a simulovat interakce běžného uživatele. Pomocí Selenia lze vyhledávat různé prvky uživatelského rozhraní uvnitř HTML dokumentu a porovnávat očekávaný výsledek aplikace se skutečnou vrácenou hodnotou. [5]
V jedné testovací třídě se může vyskytovat více oddělených testů, které se vykonávají samostatně a nezávisle na sobě. Tyto testy jsou implementovány metodami, kde každá z těchto metod je označena anotací *@Test*. Před začátkem testu se standartně volá speciální metoda *setUp()*, která má za úkol připravit prostředí na samotné provádění testu. Tato metoda se označuje anotací *@Before*.

Selenium obsahuje rozhraní *WebDriver*, díky kterému lze ovládat webové prohlížeče. Mezi nejznámější patří Goole Chrome, Mozzila Firefox a Internet Explorer. Při své práci na integračních testech jsem využil *FirefoxDriver*, který reprezentuje prohlížeč Mozzila Firefox. Jeho použití je nejjednoduší, jelikož je přímo součástí Selenia. Další objekt nezbytný pro tvorbu testů je *WebElement*, jenž reprezentuje HTML element a používá se k manipulaci s prvky uživatelského rozhraní. [6]

Při ukončení testu se stejně jako na jeho začátku volá speciální metoda *tearDown()*, která je označena anotací *@After*. Používá se pro uvolnění prostředků, které byly vyhrazeny při startu testu. Jakmile je na začátku testu *WebDriver* inicializován, můžeme na něj zavolat metody, pomocí kterých získáme přístup k elementům či dokonce informace o prohlížeči. Tyto elementy se dají získat dvěma způsoby :

1. Metoda *findElementBy(...)()*
    - Prvky objektového modelu se vyhledávají podle zakončení názvu této metody. Vyhledávat lze pomocí ID elementu, Xpath cesty, názvu tagu elementu či pomocí CSS třídy.
2. Metoda *until()*
    - Tato metoda je volána na instanci třídy *WebDriverWait*. Primární účel metody je pozastavit vykonávání testu až do chvíle, dokud není splněna nějaká podmínka (např. zobrazení elementu, na který je tato metoda volána). Defaultně má tato metoda nastavený určitý specifikovaný čas, po kterém se test ukončí. Pokud tedy uplyne časový limit metody a podmínka není splněna (element na jehož zobrazení čekáme se neukáže), test skončí neúspěchem.

Osobně jsem se na praxi setkal nejčastěji s těmito následujícími metodami :

- Metoda *getText()*
  - Vrátí text, který je uvnitř elementu.

- Metoda *click()*
  - Klikne na element.
- Metoda *findElement(By)*
  - Vyhledání určitého elementu podle jeho ID, Xpath cesty, CSS třídy nebo tagu.
- Metoda *get(String url)*
  - Otevře webovou stránku na zadané URL adrese.

4 Zadané úkoly a jejich řešení

4.1 Battery checker

Mým prvním úkolem ve firmě bylo implementovat program, který měl monitorovat aktuální stav baterie a o jeho případné změně upozornit uživatele zasláním zprávy do jeho e-malové schránky. Pracovní název tohoto programu je *Battery checker*.

4.1.1 Popis zadaného úkolu

Hlavním úkolem tohoto programu bylo v pravidelných intervalech zjišťovat, zda zařízení (v mém případě notebook) je připojen k elektrickému napájení. V situaci, že je napájení odpojeno, se měla automaticky odeslat zpráva informující uživatele o odpojení a o jeho aktuálním stavu baterie.

4.1.2 Popis řešení

Práci na tomto úkolu jsem začal hledáním vhodného řešení, jak získat informace o stavu baterie ze systému. Po zjištění, že mohu použít nativní JNA knihovnu, jsem tuto možnost použil jako primární. JNA knihovna umožňuje zjednodušený přístup k nativním metodám bez nutnosti použití JNI či nativního kódu. Pomocí této knihovny mohu jednoduše získat systémové údaje právě o stavu baterie. Sdílená knihovna, která obsahovala metody pro přístup k informacím o baterii, se jmenuje *Kernel32*. Interface, který dynamicky načítá Kernel knihovnu, lze vidět na výpisu kódu (Výpis 1).

```
public interface Kernel32 extends StdCallLibrary {

  public Kernel32 INSTANCE = (Kernel32)
     Native.loadLibrary("Kernel32", Kernel32.class);

}
```

Výpis 1: Načtení knihovny Kernel32

Interface *Kernel32* obsahuje strukturu SYSTEM_POWER_STATUS, která má public atributy *ACLineStatus* a *BatteryFlag*. Atribut *AclineStatus* vrací pomocí metody *getACLineStatusString()* řetězec **Offline**, pokud zařízení běží na baterii nebo **Online**, jestliže je napájeno z elektrické sítě. Atribut *BatteryFlag* vrací taktéž řetězec, pomocí metody *getBatteryFlagString()*, který informuje o procentuálním rozpětí aktuálního stavu baterie. Na obrázku 4 můžete vidět vzhled programu v příkazové řádce.

Kontrola aktuálního stavu baterie se provádí každou minutu, však jen pokud dojde ke změne stavu, tj. napájení je odpojeno nebo připojeno, se zašle e-mail. Tím se minimalizuje počet zbytečných zaslaných zpráv. Pro jednodušší práci s programem se nastavení e-mailu odesílatele, příjemce a hesla provádí v souboru run.bat, který je umístěn ve složce

Obrázek 4: Výpis programu ve spuštěné konzoli.

s aplikací. Vzhled tohoto souboru lze vidět na výpisu kódu (Výpis 2).

```
java -Dmail=mail@odesilatele.cz -Dpwd=hesloodesilatele
   -Dsendto=mail@prijemce.cz -jar dist/Battery_email.jar
```

Výpis 2: Vzhled spustitelného souboru run.bat

Po spuštění souboru run.bat lze vidět veškeré změny stavu baterie v konzoli. Celý program je uveden v příloze ve výpisu kódu (Výpis 4).

4.2 Tisk pomocí AdobeReaderu

Jedním z mých dalších úkolů po nástupu na stáž bylo naimplementovat řešení tisku přes spuštění procesu AdobeReader.

4.2.1 Popis zadaného úkolu

Cílem mnou napsaného programu mělo být načtení pdf dokumentu a jeho vytisknutí přes proces aplikace AdobeReader. Tisk se měl pokud možno provést na pozadí bez nutnosti zobrazení dialogu pro potvrzení tisku či náhledu pdf dokumentu.

4.2.2 Popis řešení

Práce s daným souborem měla probíhat mimo jeho lokaci, aby nedošlo k nechtěné manipulaci se souborem. Proto první úkon, který byl nutný provést, bylo vytvořit si kopii souboru do adresáře *TEMP*. Soubor jsem si otevřel v *FileInputStreamu*, který jsem následně procházel v cyklu až do jeho konce a výsledky zapsal po jednotlivých bytech do *ByteArrayOuputStreamu*. Ten jsem pomocí *FileOutputStream* a jeho metody *write()* zapsal zpět do souboru. Ten už však byl umístěn do nové lokace v adresáři *TEMP*. Zdrojový kód můžete vidět na výpisu kódu (výpis 3).

```
File file = new File("C://pdf//HelloWorld.pdf");
File fileCopy = new
    File("C://Users//HP//AppData//Local//Temp//helloworld.pdf");

FileInputStream fis = null;
FileOutputStream os = null;
try {
fis = new FileInputStream(file);
os = new FileOutputStream(fileCopy);

ByteArrayOutputStream bos = new ByteArrayOutputStream();
int cont;
byte[] buffer = new byte[1024];
while ((cont = fis.read(buffer)) != -1) {
bos.write(buffer, 0, cont);
}
os.write(bos.toByteArray());
os.flush();
os.close();
fis.close();
```

Výpis 3: Kód pro vytvoření kopie souboru do adresáře TEMP

Následně jsem se mohl pustit do práce nad kopií pdf dokumentu. Proces AdobeReader lze spustit s následujícími přepínači:

- **/n** - Pokud je již AdobeReader spuštěn s jiným pdf dokumentem, spustí separátní instanci AdobeReaderu.
- **/s** - Zabrání zobrazení úvodní obrazovky AdobeReaderu.
- **/o** - Zabrání vyskočení dialogového okna pro otevření pdf dokumentu.
- **/h** - Spustí AdobeReader ve zmenšeném okně.
- **/p** - Vytiskne požadovaný soubor.

Při vytváření požadovaného výsledku pomocí přepínačů jsem došel k tomu, že pdf dokument se sice vytiskne, avšak okno náhledu se vždy zobrazí jako zmenšenina na hlavním panelu. Je to zřejmě kvůli bezpečnosti. Na pozadí by totiž nemělo běžet nic, o čem uživatel nemá tušení.
Pro můj úkol jsem potřeboval přepínač */h* a */p*. Pro spuštění použiji třídu *Process*, kde prvním parametrem je cesta ke spustitelnému exe souboru AdobeReaderu. Poté následují přepínače, a nakonec cesta k souboru, který chceme tisknout. Cesta k souborům musí být absolutní. Celý kód programu můžete vidět v příloze výpis kódu (výpis 5).

4.3 Integrační testy

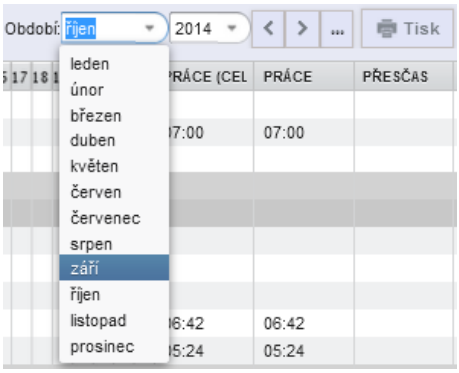
Jedním z mých dalších úkolů bylo psaní testů pro různé funkcionality jak webové aplikace, tak i mobilní pro Android platformu. Tyto testy měly za úkol simulovat běžnou interakci uživatele se systémem. Pro testování aplikací jsem použil nástroj *Selenium* a dále také framework *jUnit*. Pro popisy jednotlivých nástrojů používám jejich webové stránky. [6]

4.3.1 Popis zadaného úkolu

Každý z uživatelů firmy vlastnící docházkový systém GIRITON má možnost prohlédnout si svou docházku zpětně za jakýkoli měsíc a zobrazit či vytisknout si její grafické znázornění či celkovou sumu odpracovaných hodin. Mým úkolem bylo vytvořit integrační test na správné zobrazení náhledu jakékoli tiskové sestavy, kterou si uživatel může vytisknout. Správným zobrazením se myslí to, že se při vytváření náhledu sestavy neobjeví chybová hláška.

4.3.2 Popis řešení

Pro větší přehlednost v testu si na začátku testovací metody uložím všechny *Xpath* řetězce do proměnných typu String, které pro úspěšné dokončení testu potřebuji. Následně použiji již naimplementované metody pro přihlášení do aplikace, pro výběr agendy Docházka a pro výběr měsíce, pro který chci tisknout sestavu. Tento krok můžete vidět na obrázku 5.



Obrázek 5: ComboBox pro výběr měsíce docházky

Následně jsem použil třídu *ExpectedConditions* a její metodu *visibilityOfElementLocated(By)*. Pomocí této třídy specifikuji podmínku, která je bezpodmínečně nutná, aby test skončil s úspěchem. V mém případě to je zobrazení elementu specifikovaného pomocí proměnných s *Xpath* řetězcem. Pomocí této třídy se proklikám až k zobrazení náhledu sestavy. V konečné fázi jen zjistím, zda se nezobrazilo dialogové okno s chybovou hláškou. Finální vzhled kódu je uveden v příloze ve výpisu kódu (Výpis 6).

4.4 Vaadin - agenda Služební cesta

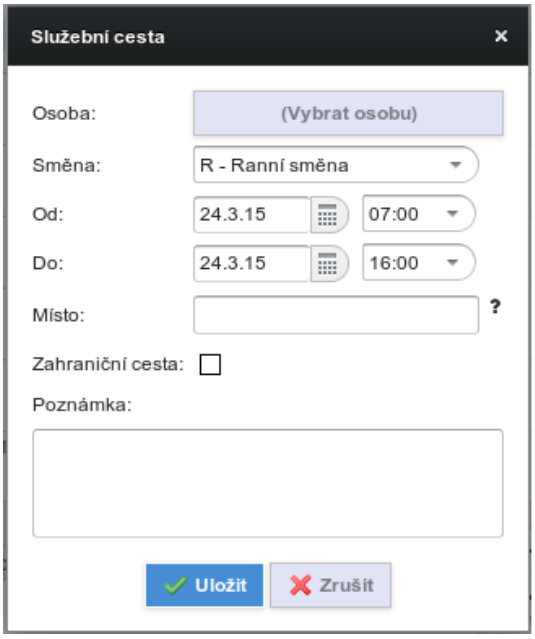
Ve Vaadin frameworku je napsán celý IS GIRITON, a proto mě práce s ním neminula. Během odborné praxe jsem pomocí Vaadinu pracoval na spoustě menších změn v aplikaci. Od změny stylu určité komponenty až po její přidání do webového rozhraní.

4.4.1 Popis zadaného úkolu

Dalším úkolem, který jsem po menších pracích s Vaadinem dostal, bylo přidání několika komponent sloužících pro rozšíření stávající agendy Služební cesta o informace o dopravě. Jejich úkolem bylo umožnit uživateli přidat nové záznamy o služebních cestách. Konkrétně jsem měl stávající dialog rozšířit o pole SPZ vozidla, cenu za litr a celkový počet litrů načerpaných pohonných hmot, místo a datum tankování, celková ujetá vzdálenost, místo výjezdu na služební cestu a nakonec informaci, zda se tankovala plná nádrž. Následně napojit jednotlivé prvky dialogu na databázi.

4.4.2 Popis řešení

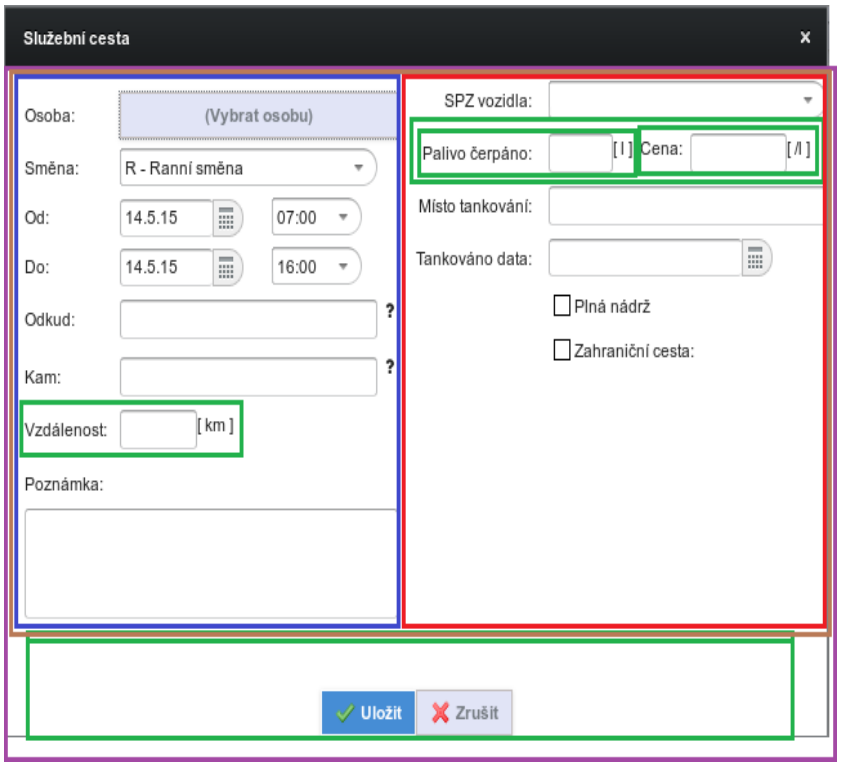
První co bylo nutné zjistit byla odpověď na otázku: „Jak se změní stávající dialogové okno po přidání nových komponent?“ Přesné rozložení bývalého okna můžete vidět na obrázku 6.



Obrázek 6: Bývalé dialogové okno agendy Služební cesta.

Po přidání nových komponent pod stávající by se dialogové okno dost vertikálně natáhlo a nevypadalo by tak uživatelsky přívětivě. Proto bylo nutné použít pro rozložení jeho částí několik druhů komponent pro vzájemné uspořádání.

- **HorizontalLayout** - komponenty, které jsou do tohoto layoutu přidány se skládají vedle sebe v pořadí, ve kterém byly přidány.
- **VerticalLayout** - stejné jako u HorizontalLayoutu jen s tím rozdílem, že komponenty se skládají pod sebe.
- **FormLayout** - rozdělí komponentu do dvou sloupců, kde v prvním sloupci je caption komponenty a ve druhém je samotná komponenta.
- **ColumnLayout** - poskytuje alternativu pro HorizontalLayout + VerticalLayout a GridLayout. Používá se při tvorbě sloupového rozvržení.
- CustomLayout - tato komponenta umožňuje vytvořit si vlastní HTML šablonu pro zobrazení.



Obrázek 7: Nové dialogové okno agendy Služební cesta.

Nejprve stávající dialog rozšířím o několik základních komponent pro interakci s uživatelem. Pro SPZ vozidla použiji *ComboBox*, který se používá pro výběr hodnoty z rozbalovací nabídky. Pro komponenty vzdálenost, místo výjezdu, načerpané palivo, cenu a místo tankování využiji *TextField* pro ruční zadání hodnot. Dále pro datum tankování použiji *PopupDateField* pro výběr data a času. V poslední řadě pro zjištění, zda uživatel tankoval plnou nádrž, použiji jednoduché políčko pro zaškrtávání - *CheckBox*.

Všechny výše zmíněné komponenty jsem použil při úpravě dialogového okna. Jak vypadá výsledné okno po úpravách můžete vidět na obrázku 7. Pro lepší orientaci a přehlednost jsou do obrázku barevně zakresleny všechny použité layout komponenty, kde zelenou barvou je vyznačen *HorizontalLayout*, fialovou barvou *VerticalLayout*, červenou barvou *FormLayout*, modrou barvou *CustomLayout* a hnědou barvou *ColumnLayout*.

5 Ostatní technologie

V této kapitole bych se chtěl věnovat ještě několika dalším nástrojům, které jsem na odborné praxi využil. Popisy nástrojů jsem převzal z jejich oficiálních webových sránek. [3]

5.1 Apache Subversion

Apache Subversion je *open source* systém používaný pro správu verzí zdrojových kódů. Byl vytvořen v roce 2000 firmou *CollabNet, Inc.* a momentálně je vyvíjen jako *Apache Software Foundation*. To umožňuje jeho bezplatné komerční použití. Subversion se vyznačuje především vysokou spolehlivostí při uchovávání cenných dat. Mezi jeho nejdůležitější vlastnosti určitě patří [4]:

- **atomické commity** - Dokud nebyl celý commit úspěšně proveden, žádná část commitovaného kódu nenabývá účinnosti.
- **řešení konfliktů** - Umožňuje řešit situace, při kterých přistupovalo k jednomu souboru více programátorů současně.
- **efektivní zpracování binárních souborů** - Subversion je stejně účinný na binární jako na textové soubory, protože používá binární algoritmus pro přenos a ukládání revizí.
- **svázání s programovacími jazyky** - Subversion API lze svázat s mnoha programovacími jazyky jako jsou Python, Perl, Java, and Ruby.
- **repozitář** - Umožňuje spravovat verze a organizovat celý projekt.

Tento nástroj jsem používal prakticky několikrát denně. Vždy když jsem potřebovat aktualizovat lokální zdrojové kódy nebo při mých lokálních změnách, které jsem chtěl nahrát na server. Pokud se při aktualizaci lokálních zdrojových kódů vyskytl konflikt (některý z kolegů pracoval na stejném souboru jako já), bylo nutné jej vyřešit. Subversion mi nabídnul pozměněné řádky či celé bloky kódů, které chci ponechat a které naopak nikoliv.

5.2 OnsenUI

OnsenUI je Javascript + CSS framework pro HTML5, PhoneGap a Cordova aplikace. Používá se pro vytváření hybridních aplikací s responzivním designem, které mohou běžet napříč platformami. OnsenUI mohou využít jak uživatelé, kteří umí jen s prostým javasctiptem, tak i uživatelé pracující s AngularJS. S dalšími verzemi tohoto nástoje se připravuje také podpora GUI nástojů jako je drag and drop. [7]

5.3 JUnit

JUnit je jednoduchý framework používaný pro jednotkové testy. Stejně jako u frameworku Selenium jsou tyto testy označovány anotací *@Test*. JUnit obsahuje několik přetížených assert metod. Na konci každého testu kontrolujeme právě pomocí assert metod, zda určitá hodnota odpovídá předpokládané. Většina vývojových systémů již obsahuje nástroje pro spuštění JUnit testů. [8]

5.4 PhoneGap

PhoneGap je open source framework sloužící pro vytváření mobilních aplikací pomocí standardizovaných webových API pro většinu platformem. Je založen na základě HTML5, CSS3 a Javascript technologiích.
Po propojení vývojového prostředí s mobilem, se při jakékoli změně ve zdrojovém kódu aplikace a jeho následném uložení automaticky pomocí tohoto nástroje zobrazí výsledek.[9]

6 Zhodnocení osobních znalostí a zkušeností

Při absolvování odborné praxe ve firmě jsem nejvíce využil znalosti z předmětu *Programovací jazyky I.*, které se přímo zabývaly programováním v jazyce Java. Taktéž jsem využil znalosti nabyté v předmětu *Základy programování* a *Algoritmy I.* a *Algoritmy II.* Tyto předměty se sice nezabývaly přímo programováním v Javě, však díky nim jsem se naučil základním dovednostem bez kterých se programátor neobejde.
Při práci na systému se mi taktéž velmi hodily znalosti nabyté v předmětu *Vývoj internetových aplikací*. Díky tomuto předmětu jsem si opět osvojil HTML, kaskádové styly, XPath a Javascript. V poslední řadě jsem měl dobrý základ znalosti SQL z předmětu *Úvod do databázových systémů* a *Databázové a informační systémy*, které jsem během praxe také využil.

6.1 Znalosti scházející v průběhu odborné praxe

V průběhu praxe jsem nejvíce postrádal znalosti frameworků, se kterými jsem se ve firmě setkal. Tyto znalosti jsem si musel doplnit především u **Vaadin** a **Selenium** frameworků, se kterými jsem přicházel do kontaktu prakticky denně. Ke konci mé stáže jsem se musel seznámit s frameworkem **OnsenUI**, pomocí kterého začala firma vytvářet mobilní multiplatformovou aplikaci s responzivním a intuitivním uživatelským rozhraním.

7 Závěr

Při absolvování odborné praxe ve firmě GIRITON jsem získal cenné zkušenosti, kterých si velmi vážím. Vyzkoušel jsem si práci v týmu programátorů na reálném Informačním systému, který používají skutečné firmy a skuteční zákazníci.
Často mi byly kolegy poskytnuty neocenitelné rady, díky kterým jsem zadaný úkol zpracoval jednodušším a rychlejším způsobem. Na závěr bych chtěl poznamenat, že pokud bych opět dostal možnost vypracovat bakalářskou práci formou praxe, určitě bych volil právě tuto skutečnost.

8 Reference

[1] GIRITON SYSTEMS S.R.O. GIRITON *[online]. 2015 [cit. 2014-05-01].* Dostupné z: http://www.giriton.cz/cz/index.html

[2] GRÖNROOS, Marko. VAADIN LTD. *Book of Vaadin: Vaadin 7 Edition - 2nd Revision [online]. 2015 [cit. 2014-05-1].* Dostupné z: https://vaadin.com/download/book-of-vaadin/vaadin-7/pdf/book-of-vaadin.pdf

[3] THE APACHE SOFTWARE FOUNDATION. *Apache Subversion [online]. 2015 [cit. 2014-05-04].* Dostupné z: http://subversion.apache.org/

[4] THE APACHE SOFTWARE FOUNDATION. *Apache Subversion Features [online]. 2015 [cit. 2014-05-04].* Dostupné z https://subversion.apache.org/features.html

[5] TESTOVÁNÍ SOFTWARU. *Selenium [online]. 2015 [cit. 2014-05-04].* Dostupné z: http://testovanisoftwaru.cz/automatizovane-testovani/selenium/

[6] SeleniumHQ. *Selenium Documentation [online]. 2015 [cit. 2014-05-04].* Dostupné z: http://docs.seleniumhq.org/docs/

[7] OnsenUI. *The Answer to Cordova UI Development [online]. 2015 [cit. 2014-05-05].* Dostupné z: http://onsen.io/

[8] JUnit *[online]. 2015 [cit. 2014-05-05].* Dostupné z: http://junit.org/

[9] PhoneGap *[online]. 2015 [cit. 2014-05-05].* Dostupné z: http://phonegap.com/

A Zdrojové kódy

```
public class Battery_email {

private static final String PROP_USERNAME = "mail";
private static final String PROP_SENDTO = "sendto";
private static final String PROP_PWD = "pwd";
private static boolean firstRun = true;
private static boolean warningEnabled;

public interface Kernel32 extends StdCallLibrary {

public Kernel32 INSTANCE = (Kernel32)
    Native.loadLibrary("Kernel32", Kernel32.class);

public class SYSTEM_POWER_STATUS extends Structure {

public byte ACLineStatus;
public byte BatteryFlag;

@Override
protected List<String> getFieldOrder() {
ArrayList<String> fields = new ArrayList<String>();
fields.add("ACLineStatus");
fields.add("BatteryFlag");
return fields;
}

public String getBatteryFlagString() {
switch (BatteryFlag) {
case (0):
return "Mezi 33% - 66%";
case (1):
return "Vyssi nez 66%";
case (2):
return "Mene nez 33%";
case (4):
return "Kriticky stav baterie";
case (8):
return "nabijeni";
case (10):
return "Mene nez 10%";
default:
return "Unknown";
}
}
```

```
public String getACLineStatusString() {
switch (ACLineStatus) {
case (0):
return "Offline";
case (1):
return "Online";
default:
return "Unknown";
}
}

@Override
public String toString() {
StringBuilder sb = new StringBuilder();
sb.append("ACLineStatus: " + getACLineStatusString() + " ");
sb.append("( " + getBatteryFlagString() + " )");
return sb.toString();
}
}

public int GetSystemPowerStatus(Kernel32.SYSTEM_POWER_STATUS
    result);
}

public static void sendSSLMessage(String flag) throws Exception {
Properties props = new Properties();
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.starttls.enable", "true");
props.put("mail.smtp.host", "smtp.gmail.com");
props.put("mail.smtp.port", "25");
props.put("mail.smtp.ssl.trust", "smtp.gmail.com");

Session session = Session.getInstance(props,
new javax.mail.Authenticator() {
@Override
protected PasswordAuthentication getPasswordAuthentication() {
return new
    PasswordAuthentication(System.getProperty(PROP_USERNAME),
    System.getProperty(PROP_PWD));
}
});
Transport transport = session.getTransport("smtps");
try {
Message message = new MimeMessage(session);
message.setFrom(new
    InternetAddress(System.getProperty(PROP_USERNAME)));
message.setRecipients(Message.RecipientType.TO,
    InternetAddress.parse(System.getProperty(PROP_SENDTO)));
message.setSubject("BATTERY");
```

```
message.setText("Napajeni bylo odpojeno. Vas NoteBook prave bezi
    na baterii."
+ "STAV:" + flag);

Transport.send(message);
System.out.println("Mail odeslan");
} finally {
transport.close();
}
}

public static void main(String[] args) throws MessagingException {

if (System.getProperty(PROP_USERNAME) == null) {
System.setProperty(PROP_USERNAME, "spactom4@gmail.com");
}
if (System.getProperty(PROP_SENDTO) == null) {
System.setProperty(PROP_SENDTO,
    System.getProperty(PROP_USERNAME));
}
if (System.getProperty(PROP_PWD) == null) {
System.setProperty(PROP_PWD, "1234567890");
}

System.out.println("Google odesilatel: " +
    System.getProperty(PROP_USERNAME));
System.out.println("Google heslo odesilatele: " +
    System.getProperty(PROP_PWD).length() + " znaku");
System.out.println("PRIJEMCE: " +
    System.getProperty(PROP_SENDTO));

int delay = 3000;
int repeat = 60 * 1000 * 1;
Timer timer = new Timer();
TimerTask task = new TimerTask() {
@Override
public void run() {
Kernel32.SYSTEM_POWER_STATUS batteryStatus = new
    Kernel32.SYSTEM_POWER_STATUS();
Kernel32.INSTANCE.GetSystemPowerStatus(batteryStatus);

if (firstRun) {
System.out.println("Beh na baterii " +
    (batteryStatus.ACLineStatus == 0) + ", batery status: " +
    batteryStatus);
}

if (batteryStatus.ACLineStatus == 0) {
if (warningEnabled) {
```

```
System.out.println("Beh na baterii, mail uz byl poslan. Batery
    status: " + batteryStatus);
return;
} else {
warningEnabled = true;
}
try {
String flag = batteryStatus.getBatteryFlagString();
sendSSLMessage(flag);
System.out.println("Beh na baterii, mail odeslan. Batery status:
    " + batteryStatus);
} catch (Exception ex) {
System.err.println("CHYBA odeslani mailu: " + ex);
}
} else {
if (warningEnabled) {
System.out.println("Beh na zasuvku. Batery satus: " +
    batteryStatus);
}
warningEnabled = false;
}
}
};

timer.scheduleAtFixedRate(task, delay, repeat);
}
}
```

Výpis 4: Program na kontrolu stavu baterie

```
public class Tisk {

private static String printername = "BrotherQL-570";
private static String drivername = "BrotherQL-570";

public static void main(String[] args) throws IOException {

File file = new File("C://pdf//HelloWorld.pdf");
String cesta = file.getAbsoluteFile().toString();
File fileCopy = new
    File("C://Users//HP//AppData//Local//Temp//helloworld.pdf");

FileInputStream fis = null;
FileOutputStream os = null;
try {
fis = new FileInputStream(file);
os = new FileOutputStream(fileCopy);

ByteArrayOutputStream bos = new ByteArrayOutputStream();
int cont;
byte[] buffer = new byte[1024];
while ((cont = fis.read(buffer)) != -1) {
bos.write(buffer, 0, cont);
}
os.write(bos.toByteArray());
os.flush();
os.close();
fis.close();

//Nasledne spusteni procesu tisku funguje pro vychozi tiskarnu
//Pokud bychom chteli vyuzit jinou nez vychozi tiskarnu, stacilo
    by odkomentovat zakomentovany radek v promene "proc"
Process proc = new ProcessBuilder(
"\"C://Program Files (x86)//Adobe//Reader
    11.0//Reader//AcroRd32.exe\"",
" /p",
" /h",
" \"C:/Users/HP/AppData/Local/Temp/helloworld.pdf\""/*,
" \"" + printername + "\"", " \"" + drivername + "\""*/).start();

//pomoci nasledujiciho bloku kodu lze zjistit nazev tiskaren
    nainstalovanych v pocitaci
/*PrintService[] printers =
    PrintServiceLookup.lookupPrintServices(null, null);
System.out.println("Pocet: " + printers.length);

for (PrintService printer : printers) {
System.out.println("Tiskarna: " + printer.getName());
```

```
//Pokud bychom chteli pouzit predem nainstalovanou tiskarnu,
   mohli bychom pouzit nasledujici uryvek kodu
if (printer.getName().equals("Brother QL-570")) {
//zde by stacilo ulozit do promene 'printname' String nazvu
   tiskarny
//uryvek kodu by se musel nachazet pred startem procesu
   AdobeReaderu
}
}*/

proc.waitFor();
} catch (Exception ex) {
ex.printStackTrace();
} finally {
try {
fis.close();
} catch (IOException ex) {
ex.printStackTrace();
}
try {
os.close();
} catch (Exception e) {
}

fileCopy.delete();
}
}
}
```

Výpis 5: Program pro tisk dokumentu pomocí AdobeReaderu

```
@Test
public void t4TiskSestava() throws Exception {
String prevMonth =
    String.format("//*[@id='dochazka-panel-date-moyepicker']/div
     [1]/div/div[3]/div/span");
String btnTisk = String.format("//*[contains(@id,
    'volbyAgendyPanel')]//*[text()[contains(.,'Tisk')]]");
String choice =
    String.format("//*[@id=\'VHasCategoryDialog-dialogTable\']//*
     [text()[contains(.,'Dochazka graficky')]]");
String btnOK =
    String.format("//*[@id='VHasCategoryDialog-btnOK']");
String finalPdf =
    String.format("//*[@id=\'VDiagReportPreview-btnClose\']");
String error = String.format("//*/div[contains(@class,
    'gwt-HTML')]");

//zde se nachazi kod pro prihlaseni do aplikace a zobrazeni
    agendy Dochazka
//zde se nachazi kod pro vyber mesice z ComboBoxu
wait.until(ExpectedConditions.visibilityOfElementLocated(
    By.xpath(prevMonth)));
driver.findElementByXPath(prevMonth).click();
wait.until(ExpectedConditions.visibilityOfElementLocated(
    By.xpath(btnTisk)));
driver.findElementByXPath(btnTisk).click();

wait.until(ExpectedConditions.visibilityOfElementLocated(
    By.xpath(choice)));
driver.findElementByXPath(choice).click();
wait.until(ExpectedConditions.visibilityOfElementLocated(
    By.xpath(btnOK)));
driver.findElementByXPath(btnOK).click();

wait.until(ExpectedConditions.visibilityOfElementLocated(
    By.xpath(btnOK)));
ElementUtils.selectTableRowCheckbox(VHasCategoryDialog.TABLE_ID,
    1, 1, driver, fluentWait);
wait.until(ExpectedConditions.visibilityOfElementLocated(
    By.xpath(btnOK)));
driver.findElementByXPath(btnOK).click();

wait.withTimeout(15,
    TimeUnit.SECONDS).until(ExpectedConditions.
     visibilityOfElementLocated(By.xpath(finalPdf)));
Assert.assertEquals("Tisk sestavy neprobehl uspesne",
    "Zavrit", driver.findElement(By.xpath(finalPdf)).getText());
try {
```

```
if (driver.findElement(By.xpath(error)) != null) {
wait.withTimeout(15,
    TimeUnit.SECONDS).until(ExpectedConditions.
     visibilityOfElementLocated(By.xpath(error)));
Assert.assertEquals("Tisk sestavy neprobehl uspesne, byla
    ZOBRAZENA chyba", "V aplikaci nastala chyba",
    driver.findElement(By.xpath(error)).getText());
}
} catch (NoSuchElementException ex) {
//Test probehl v poradku
} catch (Exception ex) {
throw new Exception(ex);
}
}
```

Výpis 6: Integrační test